**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Nguyễn Huy Tú**

**Nguyễn Duy Vũ**

**18120254**

**18120264**

**ĐỒ ÁN CUỐI KỲ**

**TRÒ CHƠI BÓNG BÀN**

**Môn học: Thực hành Phương pháp Lập trình Hướng Đối Tượng**

Thành phố Hồ Chí Minh - 2019

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**Nguyễn Huy Tú**

**Nguyễn Duy Vũ**

**18120254**

**18120264**

**ĐỒ ÁN CUỐI KỲ**

**| Đề tài |**

**TRÒ CHƠI BÓNG BÀN**

**| Giáo viên hướng dẫn |**

**ThS. Nguyễn Thành An**

**Môn học: Thực hành Phương pháp Lập trình Hướng Đối Tượng**

Thành phố Hồ Chí Minh – 2019

|  |
| --- |
| **LỜI CẢM ƠN** |

Qua thời gian học tập và rèn luyện tại trường đại học Khoa Học Tự Nhiên, Thành phố Hồ Chí Minh, được sự chỉ bảo và giảng dạy nhiệt tình của quý thầy cô khoa Công nghệ Thông tin, vốn kiến thức được tiếp thu trong quá trình học không chỉ là nền tảng cho quá trình nghiên cứu bài báo cáo mà còn là hành trang để nhóm em bước vào đời một cách vững chắc và tự tin.

Với lòng biết ơn sâu sắc nhất, nhóm xin được gửi đến quý thầy cô ở khoa Công nghệ Thông tin trường đại học Khoa Học Tự Nhiên, đặc biệt là ThS. Nguyễn Thành An, người đã tận tình hướng dẫn, giúp đỡ nhóm hoàn thành chuyên đề báo cáo này, lời cảm ơn chân thành nhất. Nếu không có những lời hướng dẫn, dạy bảo của thầy thì bài báo cáo này khó lòng mà hoàn thiện được.

Nhóm cũng muốn gửi lời cảm ơn đến gia đình, bạn bè, và người thân đã luôn ủng hộ và tạo điều kiện tốt nhất để nhóm có thể tập trung nghiên cứu đề tài này.

Trong quá trình thực tập cũng như làm báo cáo, do trình độ lý luận cũng như kinh nghiệm thực tiễn còn hạn chế nên nhóm khó tránh khỏi sai sót. Nhóm rất mong nhận được những ý kiến đóng góp quý báu của quý Thầy Cô và các bạn để nhóm học hỏi được nhiều kinh nghiệm và hoàn thành tốt hơn trong bài báo cáo sau này.

Trân trọng.

TP.HCM, ngày 15 tháng 11 năm 2019

**Sinh viên thực hiện**

|  |
| --- |
| **MỤC LỤC** |

[**LỜI CẢM ƠN** 3](#_Toc25674439)

[**MỤC LỤC** 4](#_Toc25674440)

[**BÁO CÁO ĐỒ ÁN** 5](#_Toc25674441)

[**Thông tin nhóm** 5](#_Toc25674442)

[**Thông tin đồ án** 5](#_Toc25674443)

[**Sản phẩm** 5](#_Toc25674444)

[**PHÂN CÔNG THÀNH VIÊN** 6](#_Toc25674445)

[**TIẾN TRÌNH ĐỒ ÁN** 7](#_Toc25674446)

[**PHƯƠNG THỨC & THUỘC TÍNH** 8](#_Toc25674447)

[**MÃ NGUỒN** 11](#_Toc25674448)

[**SƠ ĐỒ LỚP** 21](#_Toc25674449)

[**LƯU ĐỒ THUẬT TOÁN** 22](#_Toc25674450)

[**HƯỚNG DẪN SỬ DỤNG** 23](#_Toc25674451)

[**TÀI LIỆU THAM KHẢO** 26](#_Toc25674452)

# **BÁO CÁO ĐỒ ÁN**

### **Thông tin nhóm**

|  |  |  |  |
| --- | --- | --- | --- |
| MSSV | Họ tên | Email | Vai trò |
| 18120264 | Nguyễn Duy Vũ | Vu38988@gmail.com | Developer |
| 18120254 | Nguyễn Huy Tú | 18120254@student.hcmus.edu.vn | Developer |

### **Thông tin đồ án**

|  |  |
| --- | --- |
| Tên quy trình | Tên |
| IDE | Visual Studio 2017 |
| Báo cáo | Microsoft Word |
| Giao diện | Console Prompt/ SFML Library |
| Tên | Trò chơi bóng bàn |
| Product Owner | Thầy Nguyễn Thành An |

### **Sản phẩm**

Đồ án này viết một chương trình trò chơi bóng bàn (ping pong) trên nền console với những chức năng được yêu cầu bởi thầy Nguyễn Thành An.

# **PHÂN CÔNG THÀNH VIÊN**

Sinh viên thực hiện đồ án:

* Nguyễn Huy Tú – 18120254
* Nguyễn Duy Vũ -18120264

Phân công:

|  |  |  |
| --- | --- | --- |
| **Sinh viên thực hiện** | **Công việc** | **Nội dung** |
| Nguyễn Huy Tú | Làm các chế độ chơi (map fire, wind, rock, water). | Hiện các vật phẩm trong game. |
| Hiện các vật phẩm cộng điểm khác nhau, xuất hiện ngẫu nhiên. |
| Tăng tốc bóng, xử lý va chạm của bóng với các vật phẩm. |
| Hiển thị menu. | Thiết kế giao diện. |
| Viết báo cáo, quay video demo. | |
| Nguyễn Duy Vũ | Save game, bảng thành tích. | Lưu bảng thành tích theo thứ tự. |
| Xử lý thắng thua và kết thúc game. | Hiển thị người thắng cuộc, cho phép chọn lựa lưu game, thoát game. |
| Vẽ class diagram, thiết kế paddle tự chạy. | |

# **TIẾN TRÌNH ĐỒ ÁN**

Tiến trình đồ án dựa trên mức độ hoàn thành những chức năng được yêu cầu bởi thầy hướng dẫn.

Những câu đã làm được:

* Hiện màn hình chơi cùng với các vật phẩm của game lý (vật phẩm được xuất hiện ngẫu nhiên theo những hình học hay hình tự do).
* Xuất hiện những vật phẩm thưởng trong game theo dạng random (x2 điểm, hay x0.5 điểm, và nếu xui tạo những vật cản trước những vật phẩm)
* Save game, bảng thành tích
* Xử lý thắng thua và kết thúc game
* Tự đề xuất những ý tưởng để game chơi hấp dẫn hơn.
* Thiết kế giao diện với màu sắc đẹp
* Cho thanh truợt chạy tự động để hứng quả bóng

Những câu chưa làm được:

* Nâng cao độ khó của game khi tạo ra màn chơi có đường đến đích cho quả bóng nếu quả bóng chạm đúng điểm cần chạm.

# **PHƯƠNG THỨC & THUỘC TÍNH**

1. **Bố trí:**
2. **Chi tiết:**
   1. **Lớp Entity:**

*Các thuộc tính:*

Texture\* texture; // texture cho 1 entity

Vector2f velocity; // vận tốc của 1 entity

*Các phương thức:*

Entity(); // hàm tạo

~Entity(); // hàm hủy

void Load(string filename); // load hình cho texture

virtual void Update(); // update chuyển động cho entity

bool CheckCollision(Entity\* entity); // kiểm tra va chạm của entity này với entity được truyền vào

bool CollisionSide(Entity\* entity, COLLISION\_DIRECTION side); // check va chạm phức tạp, dùng cho fire map

Lớp **Entity** được kế thừa từ class **Sprite**, cho phép vẽ và render đồ họa các vật thể ở trong game.

Trong đó, biến texture là biến sẽ lưu trữ hình ảnh load vào vật thể, velocity là vận tốc của vật thể, vật thể sẽ di chuyển theo vận tốc này dựa vào hàm di chuyển có sẵn trong thư viện SFML.

Phương thức Entity() và ~Entity() lần lượt là hàm tạo và hàm hủy vì lớp có sử dụng con trỏ.

Phương thức Load(string filename) load hình cho các vật thể với tham số truyền vào là đường dẫn đến hình cần load.

Phương thức Update() sẽ di chuyển vật thể theo velocity đã có.

Phương thức CheckCollision(Entity\* entity) kiểm tra xem có va chạm với vật thể khác không với tham số truyền vào là 1 Entity khác. Tham số trả về là true hoặc false.

* 1. **Lớp TinyState:**

*Các phương thức:*

virtual void Initialize(RenderWindow\* window) =0;

virtual void Update(RenderWindow\* window) =0;

virtual void Render(RenderWindow\* window) =0;

virtual void Destroy(RenderWindow\* window) =0;

Lớp TinyState là lớp thuần ảo, không làm gì cả. Lớp này sẽ cho các lớp khác kế thừa và đa hình. Ý nghĩa của lớp này là thể hiện các giai đoạn trong game, mọi giai đoạn trong game đều sẽ thực hiện theo 4 bước khởi tạo, cập nhật sự kiện, render đồ họa và hủy, được thực hiện theo 4 hàm thuần ảo Initialize, Update, Render, Destroy ở trong lớp.

* 1. **Lớp Ball:**

*Các thuộc tính:*

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

Paddle\* player1; // người chơi 1

Paddle\* player2; // người chơi 2

Paddle\* rock1; // vật cản dùng cho rockmap

Paddle\* rock2; // vật cản dùng cho rockmap

Paddle\* rock3; // vật cản dùng cho rockmap

Paddle\* portal1; // portal dùng cho windmap

Paddle\* portal2; // portal dùng cho windmap

Paddle\* portal3; // portal dùng cho windmap

Paddle\* portal4; // portal dùng cho windmap

Paddle\* FireGate; // lỗ thắng dùng cho firemap

Paddle\* fire1; // vật cản dùng cho firemap

Paddle\* fire2; // vật cản dùng cho firemap

Paddle\* fire3; // vật cản dùng cho firemap

Paddle\* fire4; // vật cản dùng cho firemap

SoundBuffer\* buffer; // buffer cho âm thanh

Sound\* sound; // âm thanh

int owning; // cho biết bóng đang được sở hữu bởi paddle nào (0= không là ai cả, 1 = người chơi 1, 2 = người chơi 2)

Các biến Score1, Score2 lần lượt lưu điểm của người chơi 1, 2. Biến player1, player2 thể hiện các paddle của người chơi 1, 2. Biến rock1, rock2, rock3 là 3 vật cản được dùng trong rock map. Biến portal1, portal2, portal3, portal4 là 4 hố gió được dùng trong wind map. Biến FireGate thể hiện lỗ thắng dùng trong fire map. Biến fire1, fire2, fire3, fire4 là các vật cản sử dụng trong fire map. Buffer và sound là 2 biến được sử dụng để xử lý âm thanh trong game. Biến owning cho phép biết được người chơi nào đang sở hữu bóng.

*Các phương thức:*

Ball(Score\* Score1, Score\* Score2,Paddle\* player1, Paddle\* player2); // khởi tạo bóng

Ball(Score\* Score1, Score\* Score2, Paddle\* player1, Paddle\* player2, Paddle\* rock1, Paddle\* rock2, Paddle\* rock3); // khởi tạo bóng cho rock map

Ball(Score\* Score1, Score\* Score2, Paddle\* player1, Paddle\* player2, Paddle\* portal1, Paddle\* portal2, Paddle\* portal3, Paddle\* portal4); // khởi tạo bóng cho wind map

Ball(Score\* Score1, Score\* Score2, Paddle\* player1, Paddle\* player2, Paddle\* fire1, Paddle\* fire2, Paddle\* fire3, Paddle\* fire4, Paddle\* FireGate); // khởi tạo bóng cho fire map

~Ball();

virtual void Update(RenderWindow\* window); // update sự kiện

void RockUpdate(RenderWindow\* window); // update sự kiện cho map rock

void WindUpdate(RenderWindow\* window); // update sự kiện cho map wind

void FireUpdate(RenderWindow\* window); // update sự kiện cho fire map

void Bounce(float x, float y); // hàm nảy, dùng cho va chạm vật phức tạp (fire map)

void Reset(RenderWindow\* window); // reset (set) lại bóng, paddle

void RockReset(RenderWindow\* window); // reset (set) lại bóng, paddle, vật cản

void WindReset(RenderWindow\* window); // reset (set) lại bóng, paddle, portal

void FireReset(RenderWindow\* window); // reset (set) lại bóng, paddle, vật cản, lỗ thắng

4 phương thức tạo lập Ball ứng với 4 map khác nhau. Vì mỗi loại map khác nhau sẽ cần Ball nắm giữ những vật thể khác nhau. Hàm update là hàm virtual, kế thừa từ lớp TinyState, được dùng trong map đầu tiên, map quyết định element của game. Tiếp theo đó là các hàm update riêng biệt cho từng loại map, RockUpdate, FireUpdate, WindUpdate, WaterUpdate. Hàm Bounce là hàm di chuyển Ball theo hướng nếu như va chạm vật thể. Các hàm Reset để đặt lại vị trí các vật thể của map sau khi đã cập nhật sự kiện. Ở đây có 4 hàm reset được sử dụng cho 4 map.

* 1. **Lớp PaddlePlayer.**

*Các thuộc tính:*

int playerNumber; // 1 = người chơi 1, else = người chơi 2

Lớp PaddlePlayer kế thừa từ lớp Paddle. playerNumber cho biết paddle này là của người chơi 1 hay 2.

*Các phương thức:*

PaddlePlayer(int playerNumber); // tạo người chơi theo playerNumber

virtual void Update(); // update sự kiện

Hàm tạo paddle cho người chơi, dựa vào số được truyền vào sẽ quyết định loại paddle được tạo là gì. Hàm Update là hàm virtual từ lớp TinyState sẽ cập nhật sự kiện cho Paddle.

* 1. **Lớp PaddleAI.**

*Các thuộc tính:*

Ball\* BallObject; // track xem ball đang ở đâu

int playerNumber; // 1 = người chơi 1, else = người chơi 2

int checkrock; // dùng cho rock map, 0= khởi tạo, 1= chạm đáy, 2= chạm nóc

BallObject lưu trữ bóng, playerNumber sẽ cho biết bóng là của người chơi nào, checkrock dùng cho map rock.

*Các phương thức:*

PaddleAI(int playerNumber); // tạo người chơi theo playerNumber

void SetBall(Ball\* BallObject); // set ball cho máy

virtual void Update(); // update sự kiện

void UpdateRock(int type); // dùng cho rock map, update sự kiện cho vật cản, type = 0 đi xuống, = 1 đi lên

1. **Lớp Paddle.**

*Các thuộc tính:*

int playerNumber; // 1 = người chơi 1, else = người chơi 2

1. **Lớp Food.**

*Các thuộc tính:*

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

Ball\* BallObject; // bóng

SoundBuffer\* buffer; // buffer cho âm thanh

Sound\* sound; // âm thanh

int type; // cho biết loại food này là loại gì

bool eaten; // cho biết food đã bị ăn chưa

*Các phương thức:*

Food(Score\* Score1, Score\* Score2, Ball\* BallObject); // khởi tạo food

Food(Score\* Score1, Score\* Score2, Ball\* BallObject, int type); // khởi tạo food theo loại

~Food(); // hàm hủy

virtual void Update(); // update sự kiện

void PoisonUpdate(); // update sự kiện cho poison

void Food2Update(); // update sự kiện cho food2

void RockReset(RenderWindow\* window); // reset lại cho map rock

void WindReset(RenderWindow\* window); // reset lại cho map wind

void WaterReset(RenderWindow\* window); // reset lại cho map water

1. **Lớp Element.**

*Các thuộc tính:*

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

Ball\* BallObject; // bóng

SoundBuffer\* buffer; // buffer cho âm thanh

Sound\* sound; // âm thanh

int type; // loại nguyên tố (1= lửa, 2= nước, 3= đất, 4= gió)

*Các phương thức:*

Element(Score\* Score1, Score\* Score2,Ball\* BallObject,int type); // khởi tạo element theo loại

~Element(); // hàm hủy

virtual void Update(); // update sự kiện

void UpdateAI(); // update sự kiện

1. **Lớp FireMap.**

*Các thuộc tính:*

PaddlePlayer\* Player2; // người chơi 2

Ball\* BallObject; // bóng

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

int prevalue1; // lưu lại điểm người chơi 1 từ element state

int prevalue2; // lưu lại điểm người chơi 2 từ element state

PaddleAI\* FireGate; // lỗ thắng

PaddleAI\* Fire[3]; // vật cản

Font\* font; // font chữ

Texture texture; // background

Sprite background;

*Các phương thức:*

FireMap(int Score1, int Score2); // lấy giá trị điểm từ element state khi khởi tạo map

virtual void Initialize(RenderWindow\* window); // khởi tạo game

virtual void Update(RenderWindow\* window); // update sự kiện

virtual void Render(RenderWindow\* window); // render đồ họa

virtual void Destroy(RenderWindow\* window); // hàm hủy

void SaveGame(); // hàm lưu game

1. **Lớp RockMap.**

*Các thuộc tính:*

PaddlePlayer\* Player1; // người chơi 1

PaddlePlayer\* Player2; // người chơi 2

Ball\* BallObject; // bóng

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

int prevalue1; // lưu lại điểm người chơi 1 từ element state

int prevalue2; // lưu lại điểm người chơi 2 từ element state

PaddleAI\* Rock1; // vật cản

PaddleAI\* Rock2; // vật cản

PaddleAI\* Rock3; // vật cản

Food\* FoodObject[8]; // vật phẩm cộng điểm

Font\* font; // font chữ

Texture texture; // background

Sprite background;

*Các phương thức:*

RockMap(int Score1, int Score2); // lấy giá trị điểm từ element state khi khởi tạo map

virtual void Initialize(RenderWindow\* window); // khởi tạo game

virtual void Update(RenderWindow\* window); // update sự kiện

virtual void Render(RenderWindow\* window); // render đồ họa

virtual void Destroy(RenderWindow\* window); // hàm hủy

void SaveGame(); // hàm lưu game

1. **Lớp WindMap.**

*Các thuộc tính:*

PaddlePlayer\* Player1; // người chơi 1

PaddlePlayer\* Player2; // người chơi 2

Ball\* BallObject; // bóng

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

int prevalue1; // lưu lại điểm người chơi 1 từ element state

int prevalue2; // lưu lại điểm người chơi 2 từ element state

PaddleAI\* Portal1; // cổng gió

PaddleAI\* Portal2; // cổng gió

PaddleAI\* Portal3; // cổng gió

PaddleAI\* Portal4; // cổng gió

Food\* FoodObject[8]; // vật phẩm cộng điểm

Font\* font; // font chữ

Texture texture; // background

Sprite background;

*Các phương thức:*

WindMap(int Score1, int Score2); // lấy giá trị điểm từ element state khi khởi tạo map

virtual void Initialize(RenderWindow\* window); // khởi tạo game

virtual void Update(RenderWindow\* window); // update sự kiện

virtual void Render(RenderWindow\* window); // render đồ họa

virtual void Destroy(RenderWindow\* window); // hàm hủy

void SaveGame(); // hàm lưu game

1. **Lớp WaterMap.**

*Các thuộc tính:*

PaddlePlayer\* Player1; // người chơi 1

PaddlePlayer\* Player2; // người chơi 2

Ball\* BallObject; // bóng

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

int prevalue1; // lưu lại điểm người chơi 1 từ element state

int prevalue2; // lưu lại điểm người chơi 2 từ element state

Food\* FoodObject[4]; // vật phẩm cộng 1 điểm

Food\* Food2Object[6]; // vật phẩm cộng 2 điểm

Food\* PoisonObject[3]; // vật phẩm trừ 2 điểm

Font\* font; // font chữ

Texture texture; // background

Sprite background;

*Các phương thức:*

WaterMap(int Score1, int Score2); // lấy giá trị điểm từ element state khi khởi tạo map

virtual void Initialize(RenderWindow\* window); // khởi tạo game

virtual void Update(RenderWindow\* window); // update sự kiện

virtual void Render(RenderWindow\* window); // render đồ họa

virtual void Destroy(RenderWindow\* window); // hàm hủy

void SaveGame(); // hàm lưu game

1. **Lớp MainGame.**

*Các thuộc tính:*

PaddlePlayer\* Player1; // người chơi 1

PaddlePlayer\* Player2; // người chơi 2

Ball\* BallObject; // bóng

Score\* Score1; // điểm người chơi 1

Score\* Score2; // điểm người chơi 2

Element\* ElementObject[5];

Font\* font; // font chữ

Texture texture; // background

Sprite background;

Player1 và Player2 lưu trữ paddle của người chơi 1 và 2, BallObject là quả bóng trong trò chơi. Tiếp theo là các biến lưu điểm. ElementObject là biến quyết định nguyên tố tiếp theo của trò chơi. Font và Background là phông chữ và nền của màn chơi này.

*Các phương thức:*

virtual void Initialize(RenderWindow\* window); // khởi tạo game

virtual void Update(RenderWindow\* window); // update sự kiện

virtual void Render(RenderWindow\* window); // render đồ họa

virtual void Destroy(RenderWindow\* window); // hàm hủy

Là những hàm được kế thừa từ lớp TinyState, bao gồm hàm khởi tạo, tạo ra các chữ trên menu, hàm cập nhập sự kiện cập nhật lựa chọn người dùng. Hàm render đồ họa để vẽ các vật thể. Cuối cùng là hàm hủy khi trò chơi kết thúc

1. **Lớp MainMenu.**

*Các thuộc tính:*

Font\* font; // chọn font

Text\* play1; // text chế độ 1 người chơi

Text\* play2; // text chế độ 2 người chơi

Text\* quit; // text thoát

int selected; // lựa chọn của người chơi, =0 là 1 player, =1 là 2 playes, =2 là quit

bool upKey; // check người chơi nhấn nút lên

bool downKey; // check người chơi nhấn nút xuống

Texture texture; // background

Sprite background;

Các biến font, play1, play2 dùng để vẽ các chữ trên menu, biến selected cho biết lựa chọn của người dùng đang là gì.

*Các phương thức:*

virtual void Initialize(RenderWindow\* window); // khởi tạo menu

virtual void Update(RenderWindow\* window); // update các phím nhấn người dùng

virtual void Render(RenderWindow\* window); // render hình ảnh

virtual void Destroy(RenderWindow\* window); // hàm hủy

Là những hàm được kế thừa từ lớp TinyState, bao gồm hàm khởi tạo, tạo ra các chữ trên menu, hàm cập nhập sự kiện cập nhật lựa chọn người dùng. Hàm render đồ họa để vẽ các vật thể. Cuối cùng là hàm hủy khi trò chơi kết thúc.

# **SƠ ĐỒ LỚP**

Thầy vui lòng xem trong file ClassDiagram.pdf đính kèm.

# **LƯU ĐỒ THUẬT TOÁN**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  | | --- | | Bắt đầu |  |  | | --- | | ResizeConsole |  |  | | --- | | Menu |  |  | | --- | | Play |   true   |  | | --- | | Run |  |  | | --- | | Logic |  |  | | --- | | Draw |  |  | | --- | | Input |  |  | | --- | | Logic |   true   |  | | --- | | Quit | |

# **HƯỚNG DẪN SỬ DỤNG**

-Dùng phím W, S, Up, Down để di chuyên lựa chọn chế độ chơi (1 player, 2 player, quit) sau đó nhấn Enter

-Nhấn các phím W, S để di chuyển player 1 (paddle bên trái)

-Nhấn các phím Up, Down để di chuyển player 2(paddle bên phải)

-Nhấn Esc để lựa chọn các lựa chọn (Save game, End game, quit) sau đó nhấn Enter

+Đối với Save game chọn các lựa chọn (cotinue, 1 player, 2 player, quit) sau đó nhấn Space

+Đối với EndGame sẽ cho thấy người thắng và bảng xếp hạng điểm cao, chọn các lựa chọn (menu, quit) sau đó nhân Space

Link video demo game: *https://drive.google.com/open?id=1UMTyPyCwLnNyLqYClVcqrs3aKlcfT6xx*

# **TÀI LIỆU THAM KHẢO**

[1] NVitanovic, C++ Tutorial 18 - Simple Snake Game (Part 1), <https://www.youtube.com/watch?v=E_-lMZDi7Uw&t=126s>

[2] NVitanovic, C++ Tutorial 18 - Simple Snake Game (Part 2), <https://www.youtube.com/watch?v=W1e5wO7XR2w&t=2s>

[3] NVitanovic, C++ Tutorial 18 - Simple Snake Game (Part 3), <https://www.youtube.com/watch?v=PSoLD9mVXTA>